CPSC 340: Machine Learning and Data Mining

Fundamentals of Learning

Fall 2018

Admin

- Course webpage: www.ugrad.cs.ubc.ca/~cs340
- Assignment 1 is due Friday: you should be almost done.

- Waiting list people: you may be registered soon?
  - The other section of 340 has space.
- Graduate students who don’t need 500-level credit:
  - You should now be able to sign up for 340 (no project)?
- Auditing: message me on Piazza if you want to audit.
  - Bring your forms to me in class Friday.
  - If we can’t clear the waiting list we won’t have room for auditors.

Last Time: Supervised Learning Notation

$X =$

Egg	Milk	Fish	Wheat	Shellfish	Peanuts
0	0.7	0	0.3	0	0
0.3	0.7	0	0.6	0	0.01
0	0	0	0.8	0	0
0.3	0.7	1.2	0	0.10	0.01
0.3	0	1.2	0.3	0.10	0.01

'd' 'n'

$y =$

Sick?
1
1
0
1
1

'n'

- Feature matrix 'X' has rows as examples, columns as features.
  - x_{ij} is feature 'j' for example 'i' (quantity of food 'j' on day 'i').
  - x_i is the list of all features for example 'i' (all the quantities on day 'i').
  - x^j is column 'j' of the matrix (the value of feature 'j' across all examples).
- Label vector 'y' contains the labels of the examples.
  - y_i is the label of example 'i' (1 for "sick", 0 for "not sick").

Supervised Learning Application

- We motivated supervised learning by the “food allergy” example.

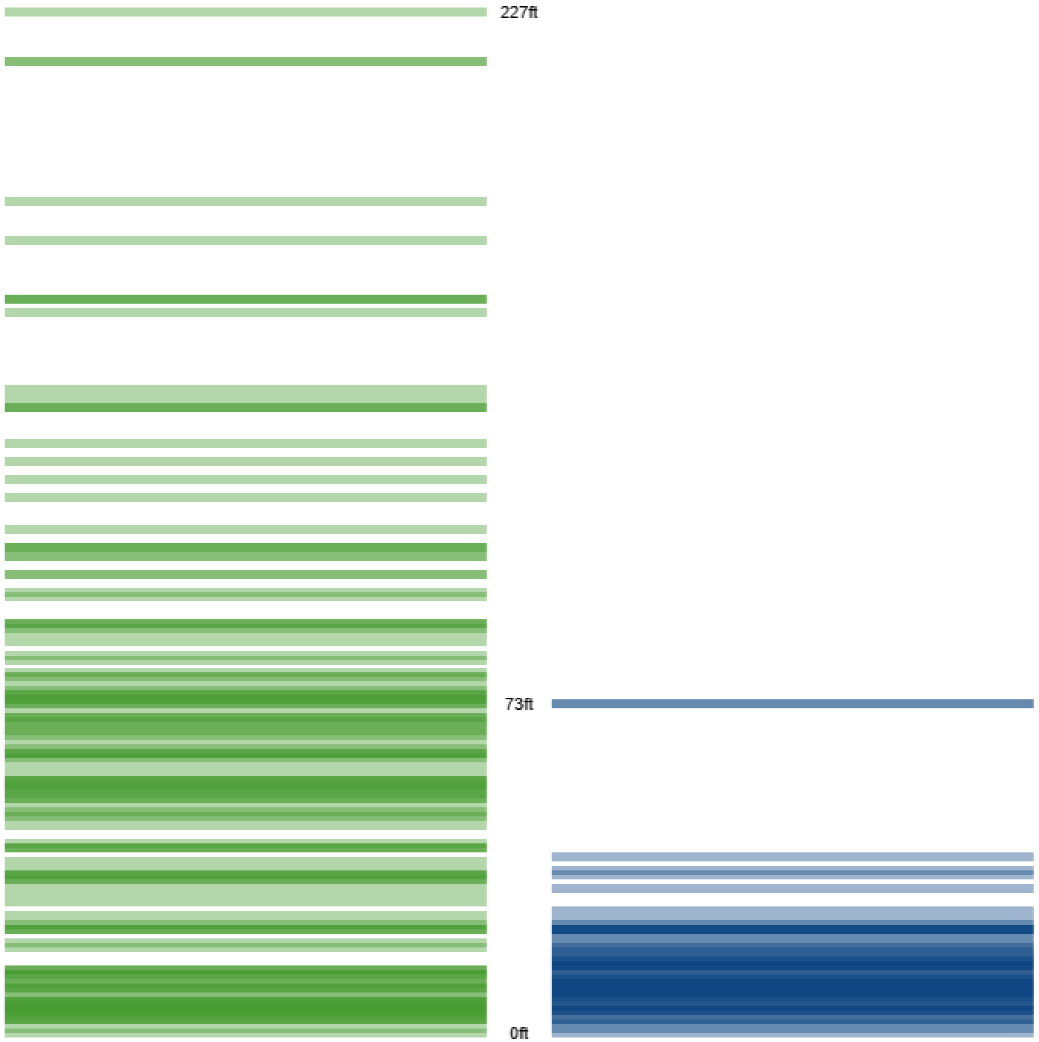
- But we can use supervised learning for any input:output mapping.
  - E-mail spam filtering.
  - Optical character recognition on scanners.
  - Recognizing faces in pictures.
  - Recognizing tumours in medical images.
  - Speech recognition on phones.
  - Your problem in industry/research?

Motivation: Determine Home City

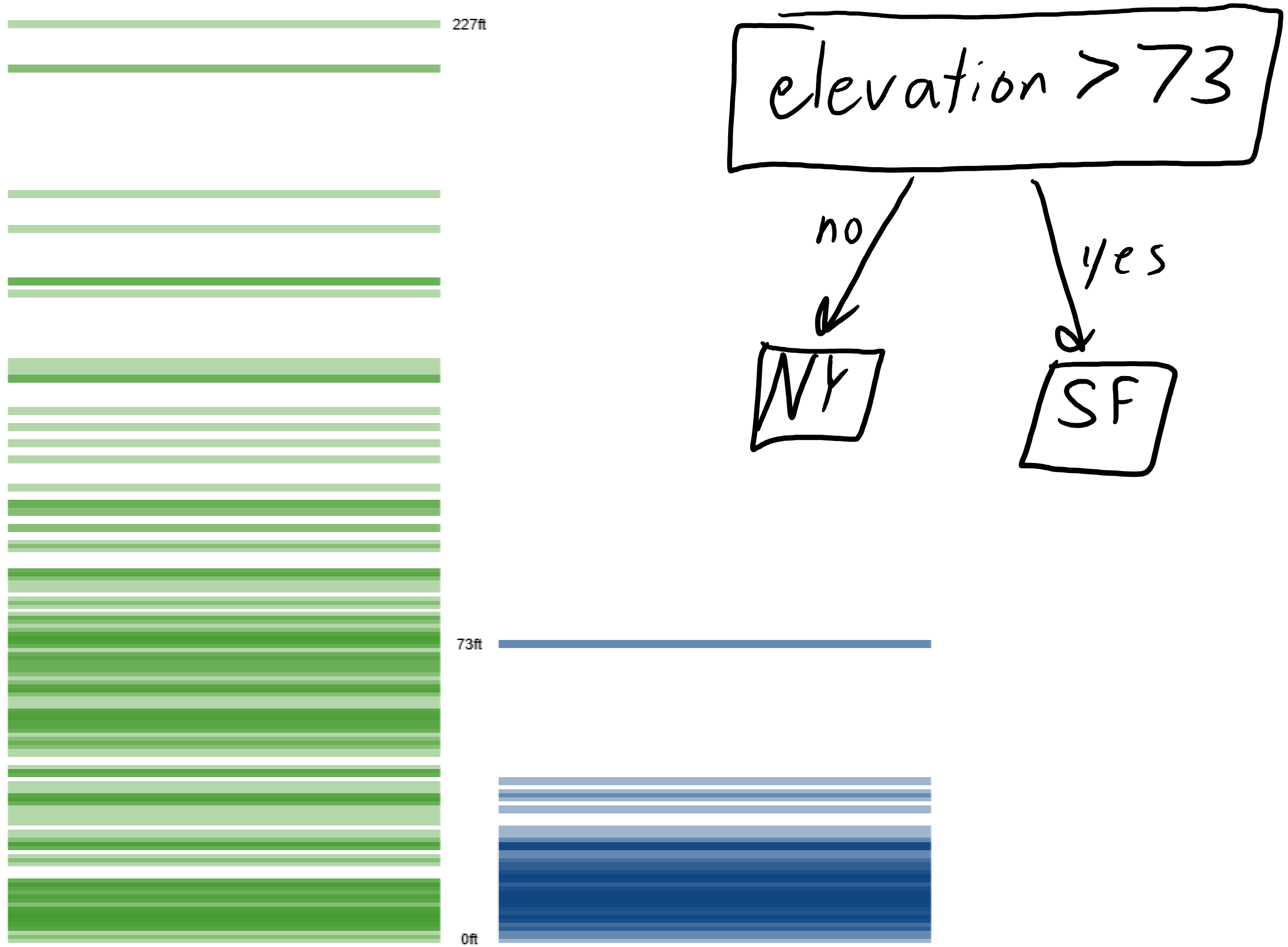
- We are given data from 248 homes.
- For each home/example, we have these features:
  - Elevation.
  - Year.
  - Bathrooms
  - Bedrooms.
  - Price.
  - Square feet.
- Goal is to build a program that predicts SF or NY.

This example and images of it come from:
http://www.r2d3.us/visual-intro-to-machine-learning-part-1

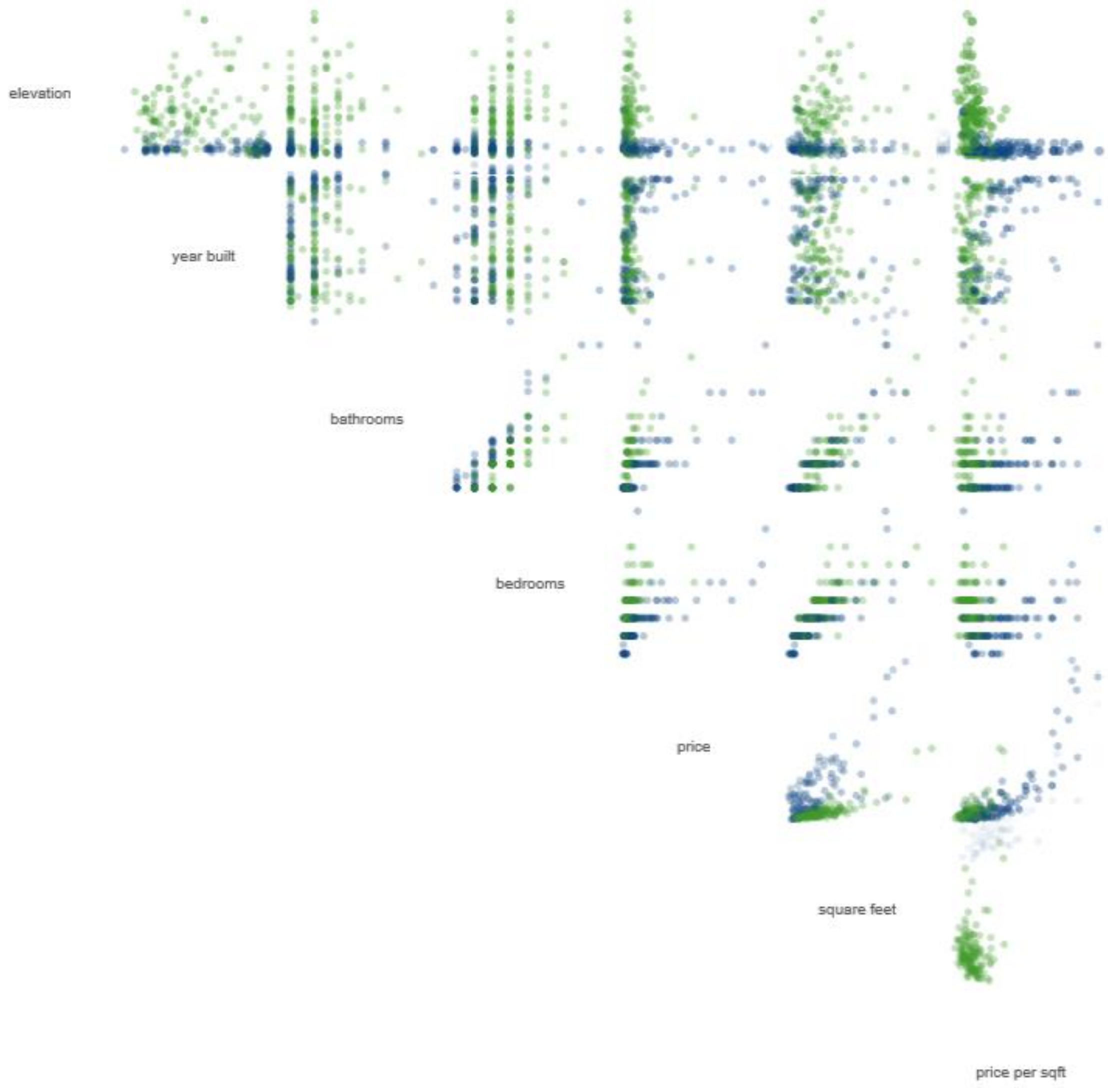
Plotting Elevation



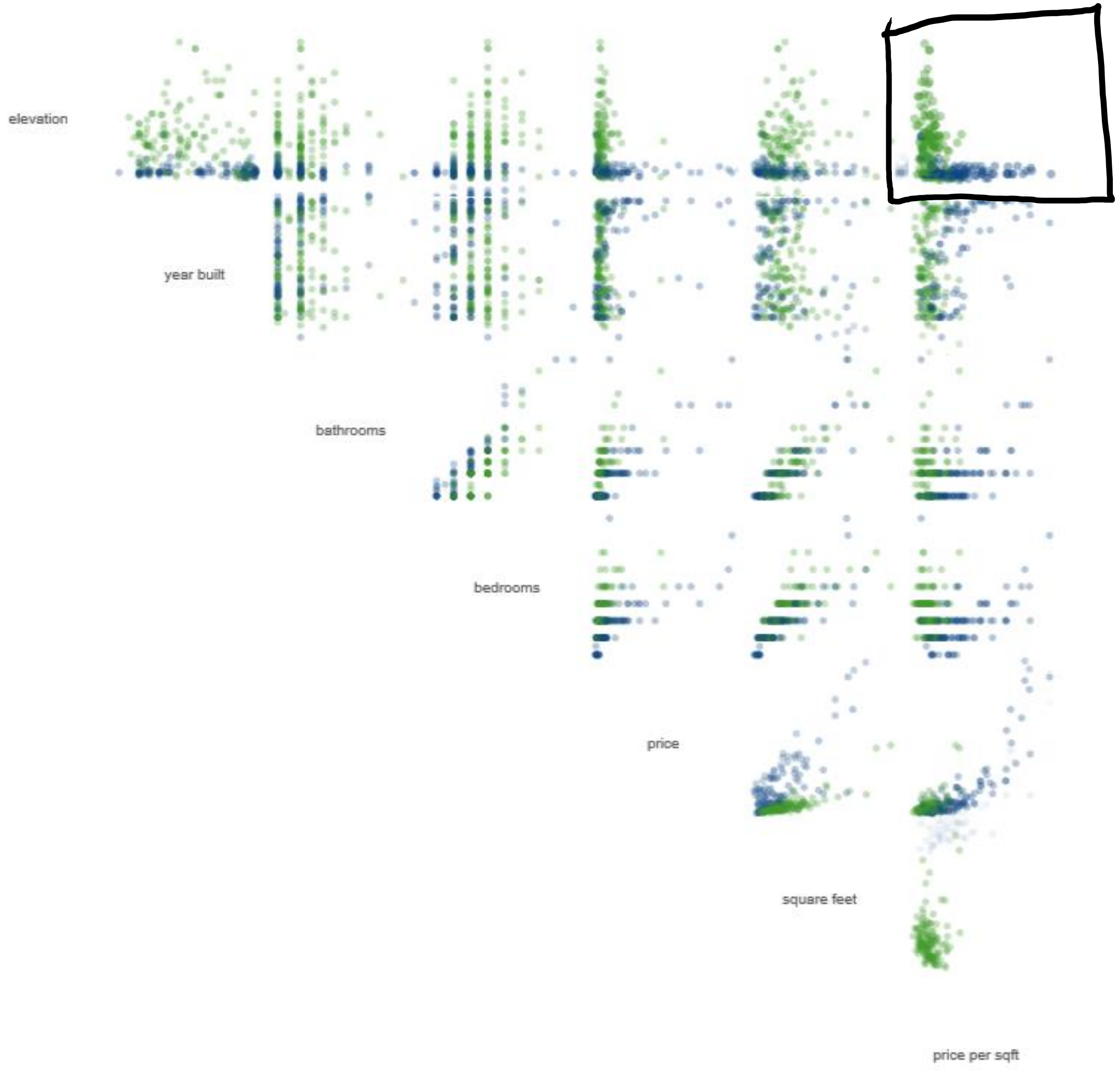
Simple Decision Stump



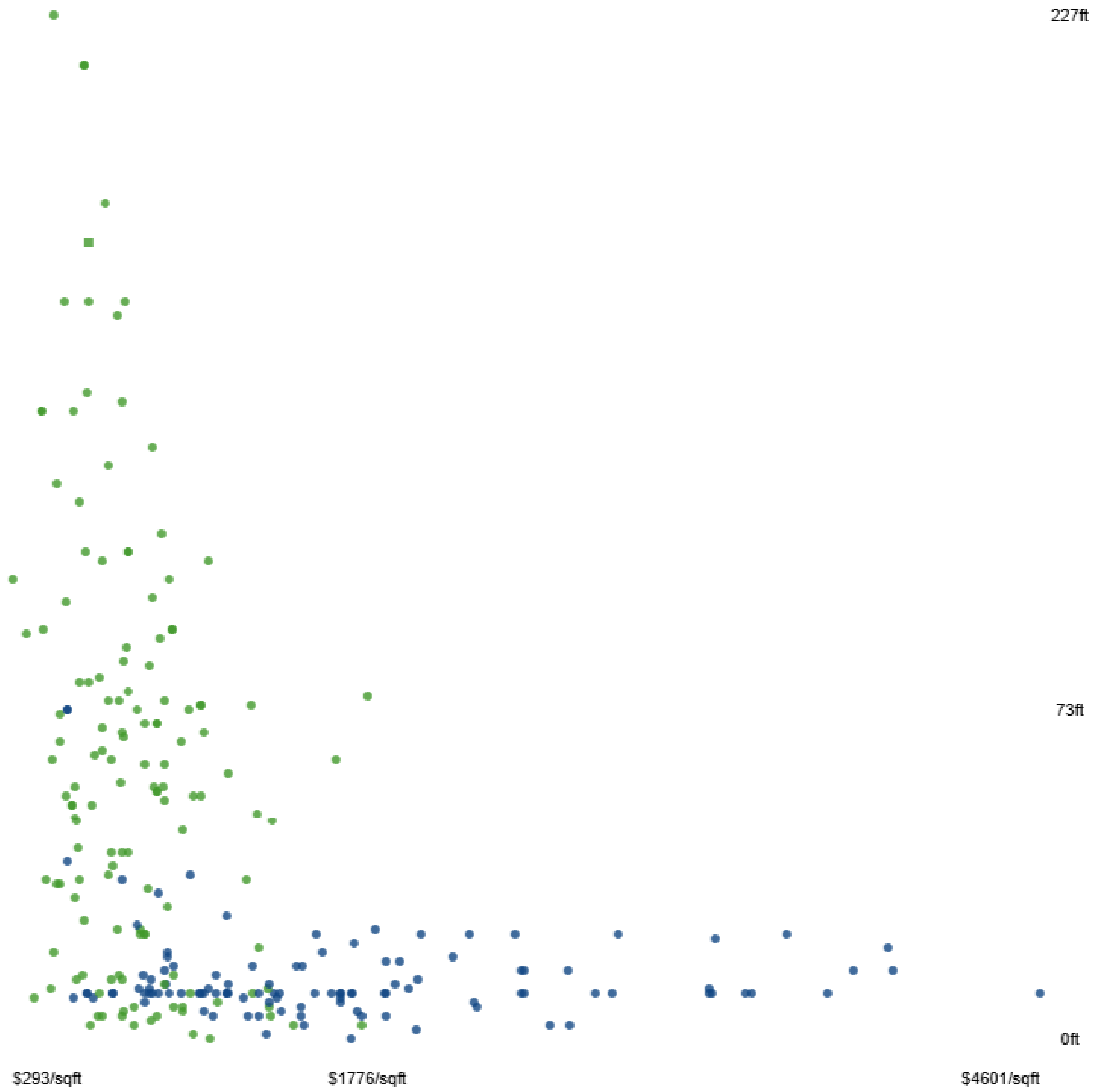
Scatterplot Array



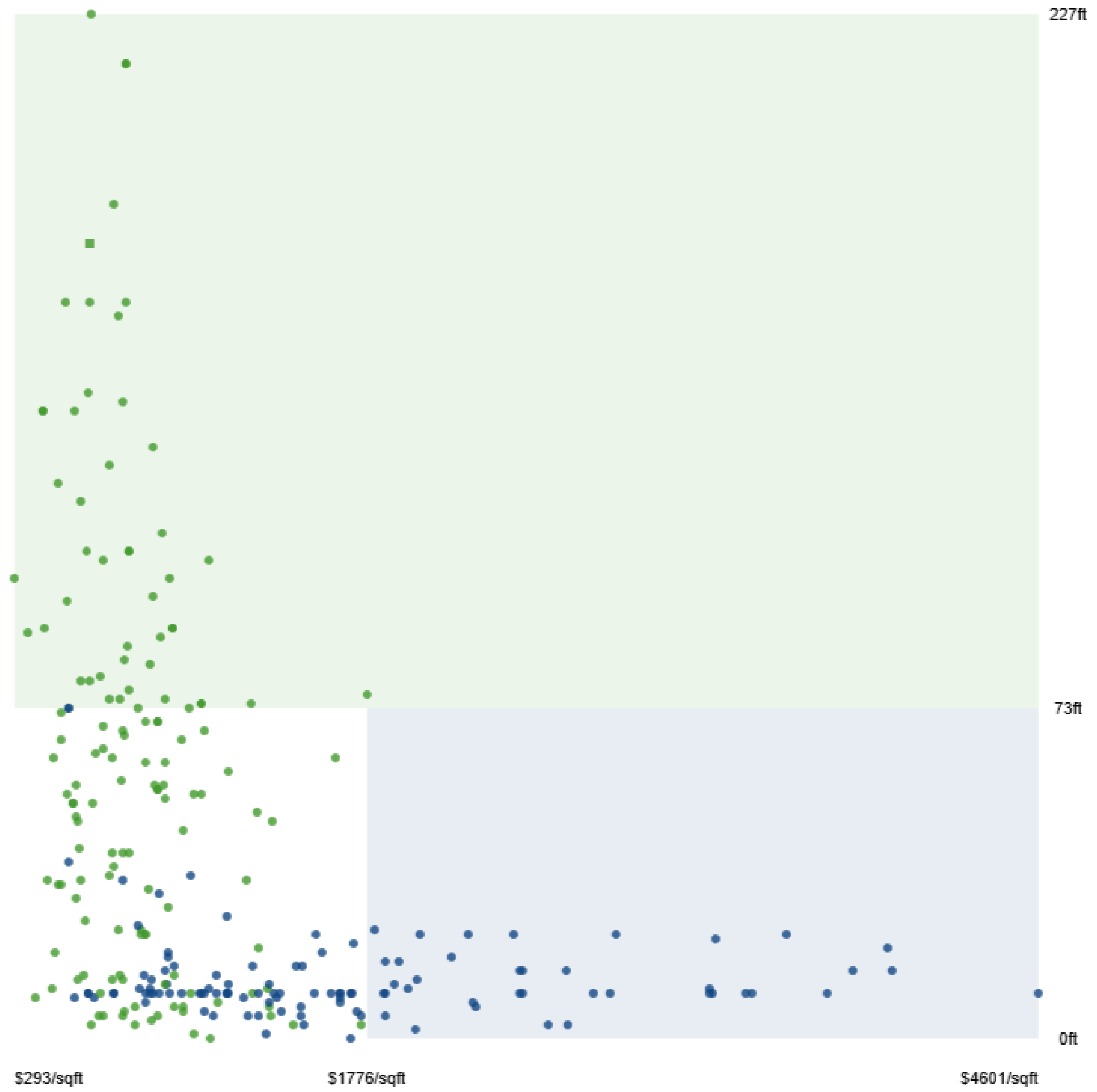
Scatterplot Array



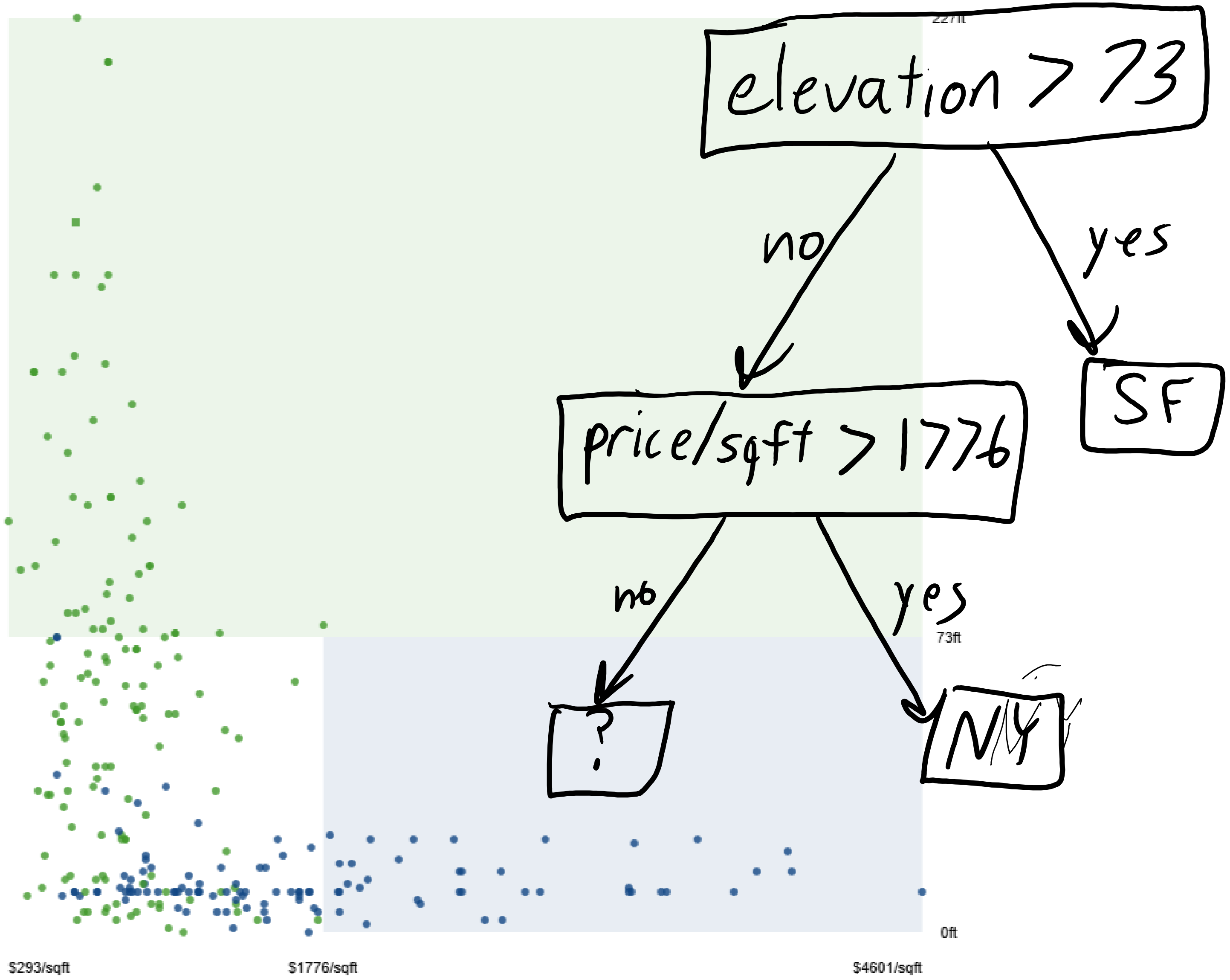
Plotting Elevation and Price/SqFt



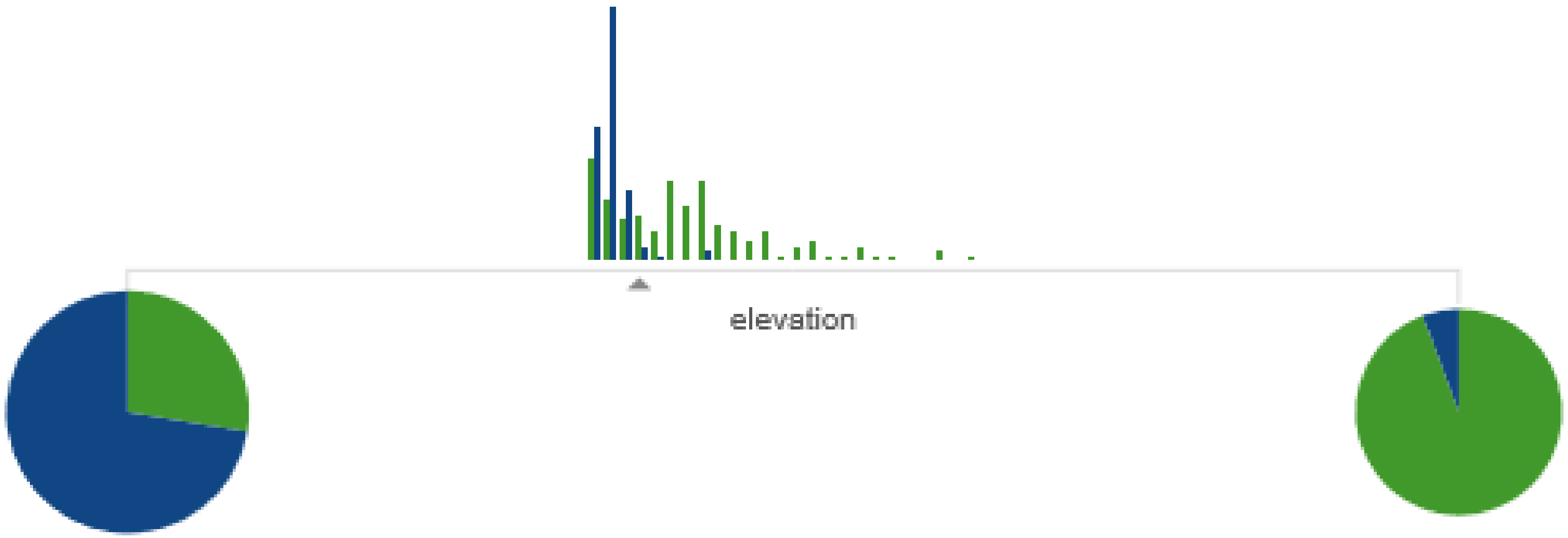
Simple Decision Tree Classification



Simple Decision Tree Classification

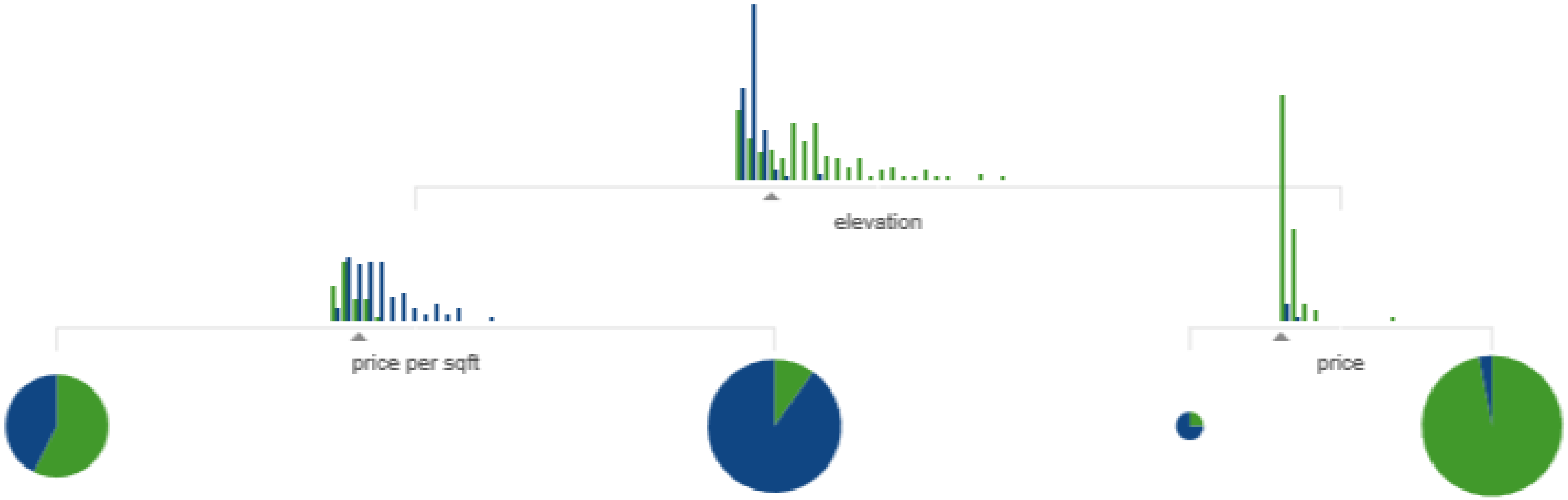


How does the depth affect accuracy?



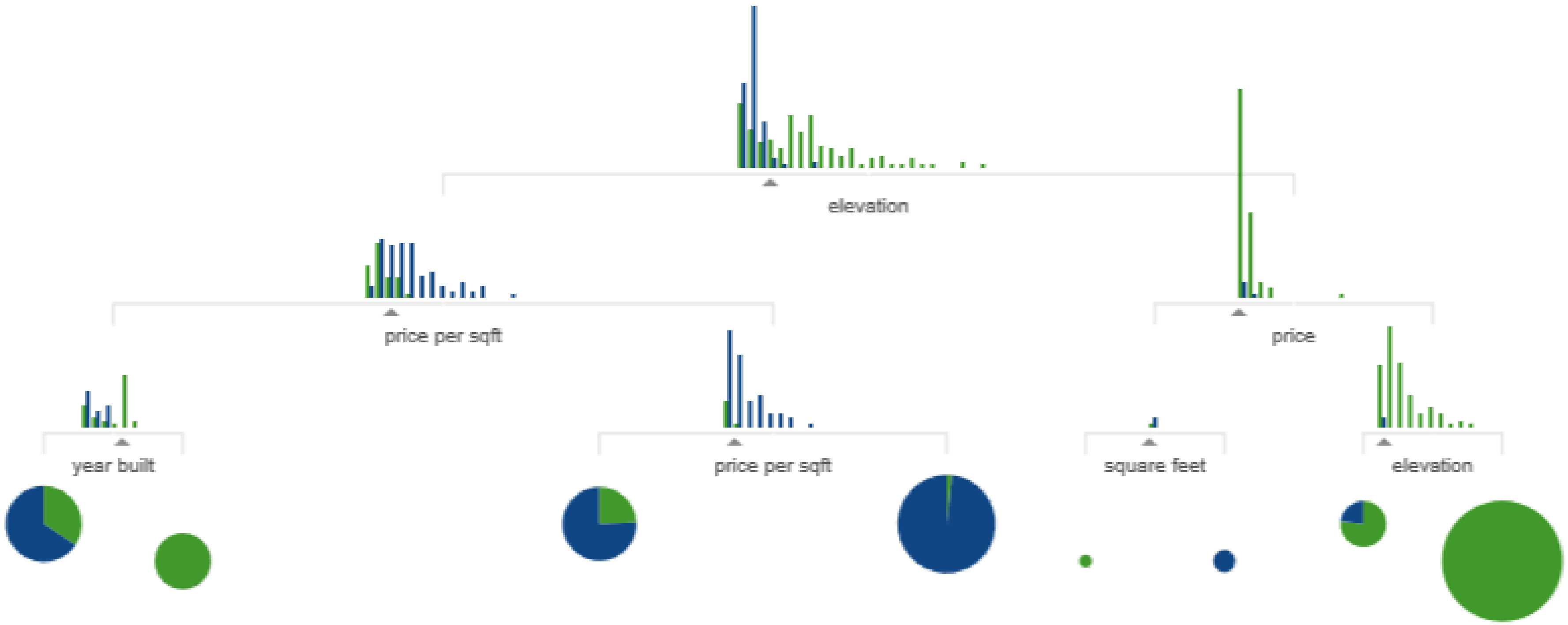
This is a good start (> 75% accuracy).

How does the depth affect accuracy?



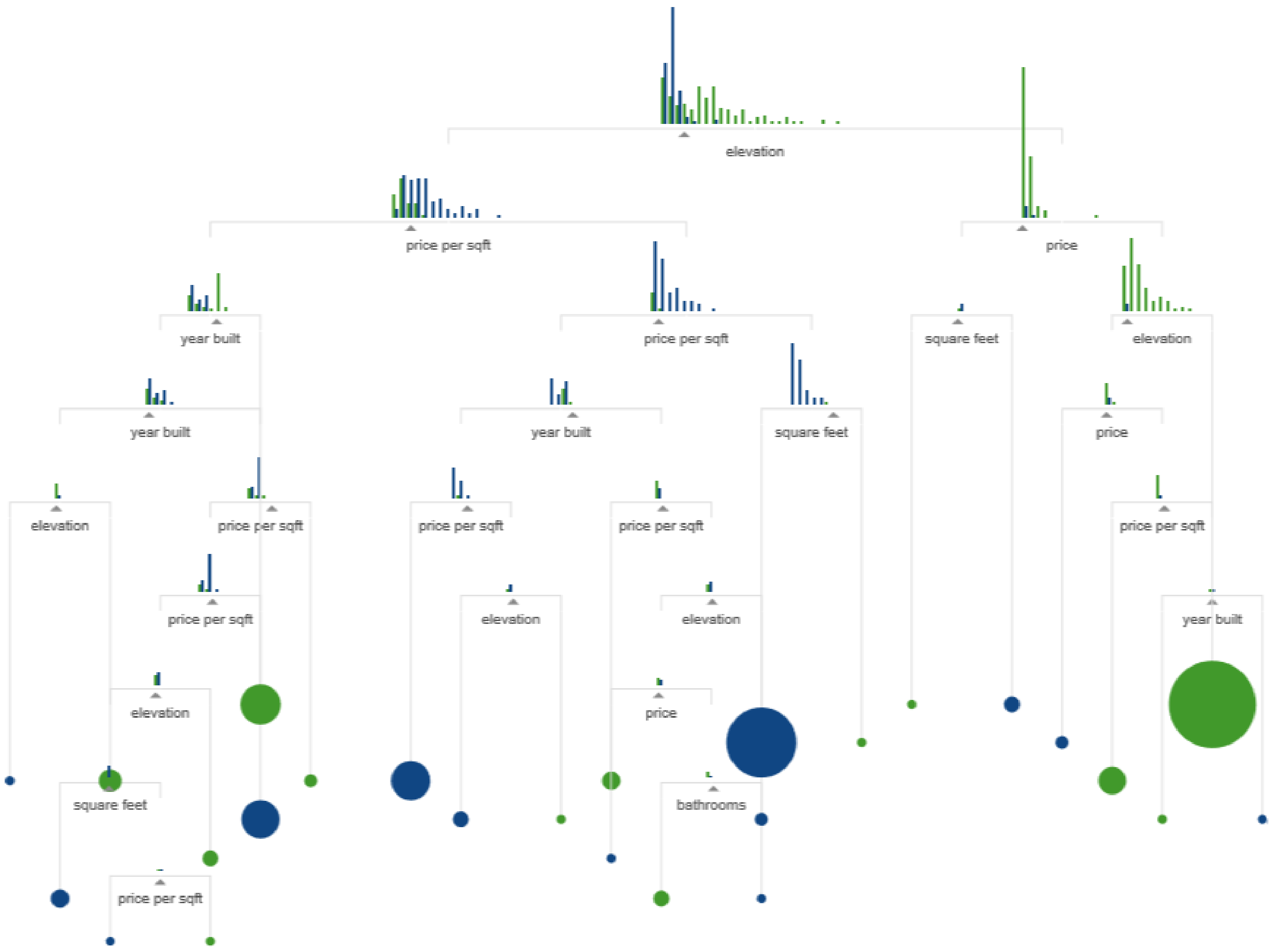
Start splitting the data recursively…

How does the depth affect accuracy?



Accuracy keeps increasing as we add depth.

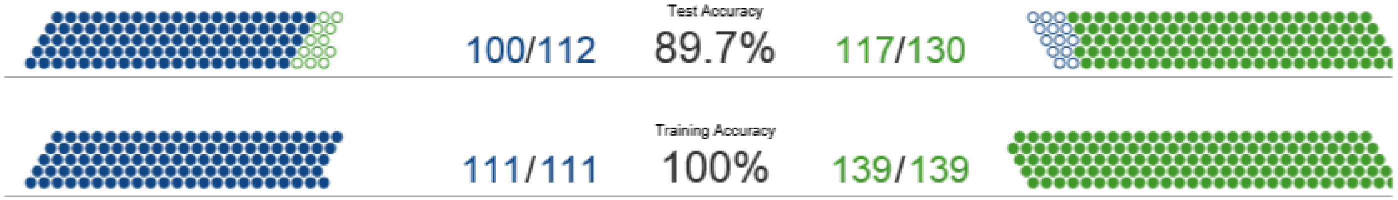
How does the depth affect accuracy?



Eventually, we can perfectly classify all of our data.

Training vs. Testing Error

- With this decision tree, 'training accuracy' is 1.
  - It perfectly labels the data we used to make the tree.
- We are now given features for 217 new homes.
- What is the 'testing accuracy' on the new data?
  - How does it do on data not used to make the tree?



- Overfitting: lower accuracy on new data.
  - Our rules got too specific to our exact training dataset.

Supervised Learning Notation

- We are given training data where we know labels:

$X =$

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…
0	0.7	0	0.3	0	0	
0.3	0.7	0	0.6	0	0.01	
0	0	0	0.8	0	0	
0.3	0.7	1.2	0	0.10	0.01	
0.3	0	1.2	0.3	0.10	0.01	

$y =$

Sick?
1
1
0
1
1

- But there is also testing data we want to label:

$\tilde{X} =$

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…
0.5	0	1	0.6	2	1	
0	0.7	0	1	0	0	
3	1	0	0.5	0	0	

$\tilde{y} =$

Sick?
?
?
?

Supervised Learning Notation

- Typical supervised learning steps:
  1. Build model based on training data X and y (training phase).
  2. Model makes predictions $\hat{y}$ on test data $\tilde{X}$ (testing phase).

- Instead of training error, consider test error:
  - Are predictions $\hat{y}$ similar to true unseen labels $\tilde{y}$?

Goal of Machine Learning

- In machine learning:
  - What we care about is the test error!
- Midterm analogy:
  - The training error is the practice midterm.
  - The test error is the actual midterm.
  - Goal: do well on actual midterm, not the practice one.
- Memorization vs learning:
  - Can do well on training data by memorizing it.
  - You’ve only learned if you can do well in new situations.

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.

- We're measuring test error to see how well we do on new data:
  - If used during training, doesn't measure this.
  - You can start to overfit if you use it during training.
  - Midterm analogy: you are cheating on the test.

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.

Tom Simonite
June 4, 2015

Why and How Baidu Cheated an Artificial Intelligence Test

Machine learning gets its first cheating scandal.

The sport of training software to act intelligently just got its first cheating scandal. Last month Chinese search company Baidu announced that its image recognition software had inched ahead of Google's on a standardized

http://www.technologyreview.com/view/538111/why-and-how-baidu-cheated-an-artificial-intelligence-test/

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.

- You also shouldn't change the test set to get the result you want.

DECEPTION AT DUKE: FRAUD IN CANCER CARE?

Were some cancer patients at Duke University given experimental treatments based on fabricated data? Scott Pelley reports.

  - http://blogs.sciencemag.org/pipeline/archives/2015/01/14/the_dukepotti_scandal_from_the_inside

https://www.cbsnews.com/news/deception-at-duke-fraud-in-cancer-care/

Digression: Golden Rule and Hypothesis Testing

- Note the golden rule applies to hypothesis testing in scientific studies.
  - Data that you collect can't influence the hypotheses that you test.

- EXTREMELY COMMON and a MAJOR PROBLEM, coming in many forms:
  - Collect more data until you coincidentally get significance level you want.
  - Try different ways to measure performance, choose the one that looks best.
  - Choose a different type of model/hypothesis after looking at the test data.

- If you want to modify your hypotheses, you need to test on new data.
  - Or at least be aware and honest about this issue when reporting results.

Digression: Golden Rule and Hypothesis Testing

- Note the golden rule applies to hypothesis testing in scientific studies.
  - Data that you collect can't influence the hypotheses that you test.

- EXTREMELY COMMON and a MAJOR PROBLEM, coming in many forms:

- References:
  - "Replication crisis in Science".
  - "Why Most Published Research Findings are False".
  - "False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant".
  - "HARKing: Hypothesizing After the Results are Known".
  - "Hack Your Way To Scientific Glory".

Is Learning Possible?

- Does training error say anything about test error?
  – In general, NO: Test data might have nothing to do with training data.
  – E.g., "adversary" takes training data and flips all labels.

X =

Egg	Milk	Fish
0	0.7	0
0.3	0.7	1
0.3	0	0

y =

Sick?
1
1
0

Xtest =

Egg	Milk	Fish
0	0.7	0
0.3	0.7	1
0.3	0	0

ytest =

Sick?
0
0
1

- In order to learn, we need assumptions:
  – The training and test data need to be related in some way.
  – Most common assumption: independent and identically distributed (IID).

IID Assumption

- Training/test data is independent and identically distributed (IID) if:
  - All examples come from the same distribution (identically distributed).
  - The example are sampled independently (order doesn't matter).

Row 1 comes from same distribution as rows 2-3.

Age	Job?	City	Rating	Income
23	Yes	Van	A	22,000.00
23	Yes	Bur	BBB	21,000.00
22	No	Van	CC	0.00
25	Yes	Sur	AAA	57,000.00

Row 4 does not depend on values in rows 1-3.

- Examples in terms of cards:
  - Pick a card, put it back in the deck, re-shuffle, repeat. → IID
  - Pick a card, put it back in the deck, repeat.
  - Pick a card, don't put it back, re-shuffle, repeat.

Not IID

IID Assumption and Food Allergy Example

- Is the food allergy data IID?
  - Do all the examples come from the same distribution?
  - Does the order of the examples matter?
- No!
  - Being sick might depend on what you ate yesterday (not independent).
  - Your eating habits might changed over time (not identically distributed).
- What can we do about this?
  - Just ignore that data isn't IID and hope for the best?
  - For each day, maybe add the features from the previous day?
  - Maybe add time as an extra feature?

Learning Theory

- Why does the IID assumption make learning possible?
  - Patterns in training examples are likely to be the same in test examples.
- The IID assumption is rarely true:
  - But it is often a good approximation.
  - There are other possible assumptions.

- Learning theory explores how training error is related to test error.
- We'll look at a simple example, using this notation:
  - E_{train} is the error on training data.
  - E_{test} is the error on testing data.

Fundamental Trade-Off

- Start with $E_{test} = E_{test}$, then add and subtract E_{train} on the right:

$$E_{test} = \underbrace{(E_{test} - E_{train})}_{\text{"approximation error"}\ E_{approx}} + \underbrace{E_{train}}_{\text{"training error"}}$$

"test error"

- How does this help?
  - If E_{approx} is small, then E_{train} is a good approximation to E_{test}.
- What does E_{approx} depend on?
  - It tends to get smaller as 'n' gets larger.
  - It tends to grow as model get more "complicated".

Fundamental Trade-Off

- This leads to a fundamental trade-off:
  1. E_{train}: how small you can make the training error.

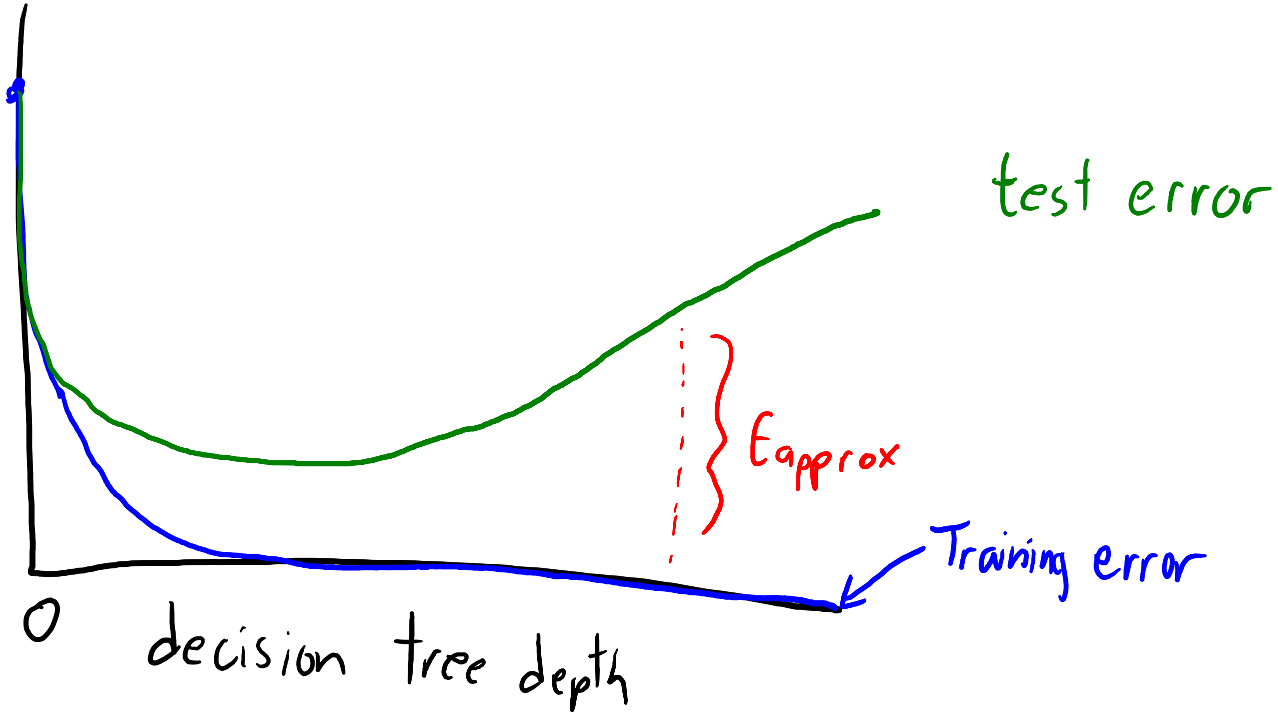
     vs.

  2. E_{approx}: how well training error approximates the test error.

- Simple models (like decision stumps):
  - E_{approx} is low (not very sensitive to training set).
  - But E_{train} might be high.
- Complex models (like deep decision trees):
  - E_{train} can be low.
  - But E_{approx} might be high (very sensitive to training set).

Fundamental Trade-Off

- Training error vs. test error for choosing depth:
  - Training error is high for low depth (underfitting)
  - Training error gets better with depth.
  - Test error initially goes down, but eventually increases (overfitting).

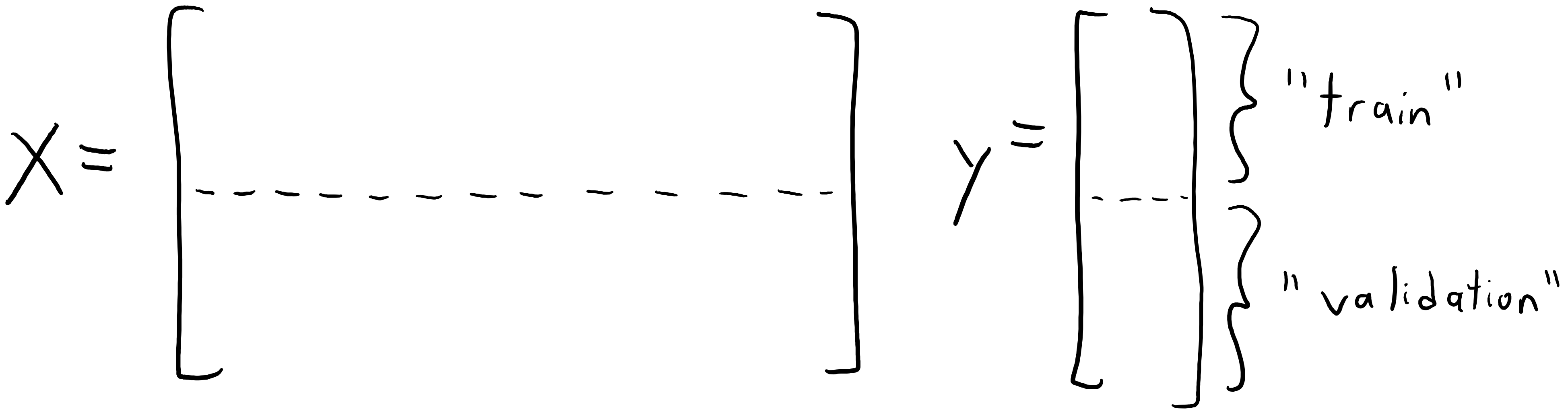


Validation Error

- How do we decide decision tree depth?
- We care about test error.
- But we can’t look at test data.
- So what do we do?????

- One answer: Use part of the training data to approximate test error.
- Split training examples into training set and validation set:
  - Train model based on the training data.
  - Test model based on the validation data.

Validation Error



Step 1 is training: model = train(X_{train}, y_{train})

Step 2 is predicting: $\hat{y}$ = predict(model, $X_{validate}$)

Step 3 is validating: error = sum($\hat{y} \neq y_{validate}$)

Note: if examples are ordered, split should be random.

Validation Error

- IID data: validation error is unbiased approximation of test error.

$$\underbrace{\mathbb{E}}_{\substack{\text{expectation}\\\text{over IID}\\\text{samples}}}[\underbrace{E_{valid}}_{\substack{\text{validation}\\\text{error}}}] = \underbrace{\mathbb{E}}_{\substack{\text{expectation}\\\text{over IID}\\\text{samples}}}[\underbrace{E_{test}}_{\substack{\text{test}\\\text{error}}}]$$

- Midterm analogy:
  - You have 2 practice midterms.
  - You hide one midterm, and spend a lot of time working through the other.
  - You then do the other practice term, to see how well you'll do on the test.

- We typically use validation error to choose "hyper-parameters"…

Notation: Parameters and Hyper-Parameters

- The decision tree rule values are called "parameters".
  - Parameters control how well we fit a dataset.
  - We "train" a model by trying to find the best parameters on training data.

- The decision tree depth is a called a "hyper-parameter".
  - Hyper-parameters control how complex our model is.
  - We can't "train" a hyper-parameter.
    - You can always fit training data better by making the model more complicated.
  - We "validate" a hyper-parameter using a validation score.

Choosing Hyper-Parameters with Validation Set

- So to choose a good value of depth ("hyper-parameter"), we could:
  - Try a depth-1 decision tree, compute validation error.
  - Try a depth-2 decision tree, compute validation error.
  - Try a depth-3 decision tree, compute validation error.
  - …
  - Try a depth-20 decision tree, compute validation error.
  - Return the depth with the lowest validation error.

- After you choose the hyper-parameter, we usually re-train on the full training set with the chosen hyper-parameter.

Digression: Optimization Bias

- Another name for overfitting is "optimization bias":
  - How biased is an "error" that we optimized over many possibilities?

- Optimization bias of parameter learning:
  - During learning, we could search over tons of different decision trees.
  - So we can get "lucky" and find one with low training error by chance.
    - "Overfitting of the training error".

- Optimization bias of hyper-parameter tuning:
  - Here, we might optimize the validation error over 20 values of "depth".
  - One of the 20 trees might have low validation error by chance.
    - "Overfitting of the validation error".

Digression: Example of Optimization Bias

- Consider a multiple-choice (a,b,c,d) “test” with 10 questions:
  – If you choose answers randomly, expected grade is 25% (no bias).
  – If you fill out two tests randomly and pick the best, expected grade is 33%.
    - Optimization bias of ~8%.
  – If you take the best among 10 random tests, expected grade is ~47%.
  – If you take the best among 100, expected grade is ~62%.
  – If you take the best among 1000, expected grade is ~73%.
  – If you take the best among 10000, expected grade is ~82%.
    - You have so many “chances” that you expect to do well.

- But on new questions the “random choice” accuracy is still 25%.

Factors Affecting Optimization Bias

- If we instead used a 100-question test then:
  - Expected grade from best over 1 randomly-filled test is 25%.
  - Expected grade from best over 2 randomly-filled test is ~27%.
  - Expected grade from best over 10 randomly-filled test is ~32%.
  - Expected grade from best over 100 randomly-filled test is ~36%.
  - Expected grade from best over 1000 randomly-filled test is ~40%.
  - Expected grade from best over 10000 randomly-filled test is ~47%.

- The optimization bias grows with the number of things we try.
  - "Complexity" of the set of models we search over.
- But, optimization bias shrinks quickly with the number of examples.
  - But it's still non-zero and growing if you over-use your validation set!

Summary

- Training error vs. testing error:
  - What we care about in machine learning is the testing error.
- Golden rule of machine learning:
  - The test data cannot influence training the model in any way.
- Independent and identically distributed (IID):
  - One assumption that makes learning possible.
- Fundamental trade-off:
  - Trade-off between getting low training error and having training error approximate test error.
- Validation set:
  - We can save part of our training data to approximate test error.
- Hyper-parameters:
  - Parameters that control model complexity, typically set with a validation set.

- Next time:
  - We discuss the “best” machine learning method.

Bounding E_{approx}

- Let's assume we have a fixed model 'h' (like a decision tree), and then we collect a training set of 'n' examples.
- What is the probability that the error on this training set (E_{train}), is within some small number ε of the test error (E_{test})?

- From "Hoeffding's inequality" we have:

$$P\left[\,|E_{train}(h) - E_{test}(h)| > \varepsilon\,\right] \leq 2\exp(-2\varepsilon^2 n)$$

- This is great! In this setting the probability that our training error is far from our test error goes down exponentially in terms of the number of samples 'n'.

Bounding E_{approx}

- Unfortunately, the last slide gets it backwards:
  - We usually don't pick a model and then collect a dataset.
  - We usually collect a dataset and then pick the model 'w' based on the data.
- We now picked the model that did best on the data, and Hoeffding's inequality doesn't account for the optimization bias of this procedure.
- One way to get around this is to bound (E_{test} – E_{train}) for *all* models in the space of models we are optimizing over.
  - If bound it for all models, then we bound it for the best model.
  - This gives looser but correct bounds.

Bounding E_{approx}

- If we only optimize over a finite number of events ‘k’, we can use the “union bound” that for events $\{A_1, A_2, \ldots, A_k\}$ we have:

$$p(A_1 \cup A_2 \cup \cdots \cup A_k) \leq \sum_{i=1}^{k} p(A_i)$$

- Combining Hoeffding’s inequality and the union bound gives:

$$p(|E_{train}(h_1) - E_{test}(h_1)| > \varepsilon \ \cup \ |E_{train}(h_2) - E_{test}(h_2)| > \varepsilon \ \cup \cdots \cup \ |E_{train}(h_k) - E_{test}(h_k)| > \varepsilon)$$

$$\leq \sum_{i=1}^{k} p(|E_{train}(h_i) - E_{test}[h_i]| > \varepsilon)$$

$$\leq \sum_{i=1}^{k} 2\exp(-2\varepsilon^2 n)$$

$$\leq 2k\exp(-2\varepsilon^2 n)$$

Bounding E_{approx}

- So, with the optimization bias of setting "h*" to the best 'h' among 'k' models, probability that (Etest – Etrain) is bigger than ε satisfies:

$$p(|E_{train}(h^*) - E_{test}(h^*)| > \epsilon) \leq 2k\exp(-2\epsilon^2 n)$$

- So optimizing over a few models is ok if we have lots of examples.
- If we try lots of models then (E_{test} – E_{train}) could be very large.
- Later in the course we'll be searching over continuous models where k = infinity, so this bound is useless.
- To handle continuous models, one way is via the VC-dimension.
  – Simpler models will have lower VC-dimension.

Refined Fundamental Trade-Off

- Let E_{best} be the irreducible error (lowest possible error for *any* model).
  - For example, irreducible error for predicting coin flips is 0.5.
- Some learning theory results use E_{best} to futher decompose E_{test}:

$$E_{test} = \underbrace{(E_{test} - E_{train})}_{\text{"variance"}} + \underbrace{(E_{train} - E_{best})}_{\text{"bias"}} + \underbrace{E_{best}}_{\text{"noise"}}$$

- This is similar to the bias-variance decomposition:
  - Term 1: measure of variance (how sensitive we are to training data).
  - Term 2: measure of bias (how low can we make the training error).
  - Term 3: measure of noise (how low can any model make test error).

Refined Fundamental Trade-Off

- Decision tree with high depth:
  - Very likely to fit data well, so bias is low.
  - But model changes a lot if you change the data, so variance is high.
- Decision tree with low depth:
  - Less likely to fit data well, so bias is high.
  - But model doesn’t change much you change data, so variance is low.

- And degree does not affect irreducible error.
  - Irreducible error comes from the best possible model.

Bias-Variance Decomposition

- You may have seen "bias-variance decomposition" in other classes:
  - Assumes $\tilde{y}_i = \bar{y}_i + \varepsilon$, where ε has mean 0 and variance σ^2.
  - Assumes we have a "learner" that can take 'n' training examples and use these to make predictions $\hat{y}_i$.
- Expected squared test error in this setting is

$$\underbrace{\mathbb{E}[(\tilde{y}_i - \hat{y}_i)^2]}_{\text{"test squared error"}} = \underbrace{\mathbb{E}[(\hat{y}_i - \bar{y}_i)]^2}_{\text{"bias"}} + \underbrace{(\mathbb{E}[\hat{y}_i^2] - \mathbb{E}[\hat{y}_i]^2)}_{\text{"variance"}} + \underbrace{\sigma^2}_{\text{"noise"}}$$

  - Where expectations are taken over possible training sets of 'n' examples.
  - Bias is expected error due to having wrong model.
  - Variance is expected error due to sensitivity to the training set.
  - Noise (irreducible error) is the best can hope for given the noise (E_{best}).

Bias-Variance vs. Fundamental Trade-Off

- Both decompositions serve the same purpose:
  - Trying to evaluate how different factors affect test error.

- They both lead to the same 3 conclusions:
  1. Simple models can have high E_{train}/bias, low E_{approx}/variance.
  2. Complex models can have low E_{train}/bias, high E_{approx}/variance.
  3. As you increase 'n', E_{approx}/variance goes down (for fixed complexity).

Bias-Variance vs. Fundamental Trade-Off

- So why focus on fundamental trade-off and not bias-variance?
  - Simplest viewpoint that gives these 3 conclusions.
  - No assumptions like being restricted to squared error.

  - You can measure E_{train} but not E_{approx} (1 known and 1 unknown).
    - If E_{train} is low and you expect E_{approx} to be low, then you are happy.
      - E.g., you fit a very simple model or you used a huge independent validation set.

  - You can't measure bias, variance, or noise (3 unknowns).
    - If E_{train} is low, bias-variance decomposition doesn't say anything about test error.
      - You only have your training set, not distribution over possible datasets.
      - Doesn't say if high E_{test} is due to bias or variance or noise.

Learning Theory

- Bias-variance decomposition is a bit weird compared to our previous decompositions of E_{test}:
  - Bias-variance decomposition considers expectation over *possible training sets*.
  - But doesn't say anything about test error with *your* training set.

- Some keywords if you want to learn about learning theory:
  - Bias-variance decomposition, sample complexity, probably approximately correct (PAC) learning, Vapnik-Chernovenkis (VC) dimension, Rademacher complexity.

- A gentle place to start is the "Learning from Data" book:
  - https://work.caltech.edu/telecourse.html

A Theoretical Answer to "How Much Data?"

- Assume we have a source of IID examples and a fixed class of parametric models.
  - Like "all depth-5 decision trees".
- Under some nasty assumptions, with 'n' training examples it holds that:

  E[test error of best model on training set] – (best test error in class) = O(1/n).

- You rarely know the constant factor, but this gives some guidelines:
  - Adding more data helps more on small datasets than on large datasets.
    - Going from 10 training examples to 20, difference with best possible error gets cut in half.
      - If the best possible error is 15% you might go from 20% to 17.5% (this does **not** mean 20% to 10%).
    - Going from 110 training examples to 120, error only goes down by ~10%.
    - Going from 1M training examples to 1M+10, you won't notice a change.
  - Doubling the data size cuts the error in half:
    - Going from 1M training to 2M training examples, error gets cut in half.
    - If you double the data size and your test error doesn't improve, more data might not help.

Can you test the IID assumption?

- In general, testing the IID assumption is not easy.
  – Usually, you need background knowledge to decide if it's reasonable.

- Some tests do exist, like shuffling the order of data and then measuring if some basic statistics agree.
  – It's reasonable to check if summary statistics of train and test data agree.
    - If not, your trained model may not be so useful.

- Some discussion here:
  – https://stats.stackexchange.com/questions/28715/test-for-iid-sampling